Του Γιώργου Καργιωτάκη <kargig@gmail.com>

Ο Γιώργος δουλεύει σε μία ελληνική εταιρεία που φτιάχνει Linux based xDSL routers και ασχολείται με θέματα QoS, VoIP και IPv6 σε Linux!

Κατακτήστε το Z shell

Το Zsh είναι ένα πολύ ισχυρό shell κατάλληλο για ανθρώπους που χρησιμοποιούν συχνά τη γραμμή εντολών.

void.gr/kargig

Οσοι χρησιμοποιούν συχνά τη γραμμή εντολών, γνωρίζουν πόσο δυνατό και ευέλικτο εργαλείο μπορεί να γίνει. Σε αυτό το άρθρο θα γίνει μία μικρή εισαγωγή σε κάποιες από τις δυνατότητες του zsh, το οποίο είναι ένα από τα πιο ισχυρά shells που υπάρχουν.

Το zsh είναι ένα διαδραστικό shell (κέλυφος), καθώς και μία πολύ ισχυρή γλώσσα scripting. Συνήθως τα shells ανήκουν σε δύο κατηγορίες, στα Bourne-like shells (bash, dash κ.ά.) και στα C-like shells (tcsh) – τα τελευταία ξεχωρίζουν αρκετά, μια και η σύνταξή τους μοιάζει με αυτή της γλώσσας C. Τα Bourne shells είναι αρκετά πιο εύκολα στη χρήση από τα C shells, αλλά συχνά η σύνταξη ενός καλού shell script καταλήγει να γίνει αρκετά περίπλοκη, εξαιτίας της σύνταξης σε σχέση με τα C shells. Ως επέκταση του Bourne shell δημιουργήθηκε το Korn shell (ksh), το οποίο διαθέτει την ευκολία χρήσης του Bourne, έχοντας προσθέσει επεκτάσεις για καλύτερο έλεγχο των διεργασιών, πολλαπλές διεργασίες που τρέχουν στο background, δυνατότητα μορφοποίησης προηγούμενων εντολών που έχουν τρέξει από τη γραμμή εντολών, καθώς και λειτουργίες από τα C shells που καθιστούν πιο εύκολο τον προγραμματισμό. Το zsh είναι περισσότερο επικεντρωμένο στη χρήση από τη γραμμή εντολών και λιγότερο στον προγραμματισμό και βοηθά πάρα πολύ όσους χρησιμοποιούν καθημερινά τη γραμμή εντολών.

Η εγκατάσταση του zsh γίνεται μέσω του πακέτου zsh στις περισσότερες διανομές: aptitude install zsh για Debian/Ubuntu, yum install zsh για το Fedora κ.λπ.

Το zsh είναι περισσότερο επικεντρωμένο στη χρήση από τη γραμμή εντολών και λιγότερο στον προγραμματισμό και βοηθά πάρα πολύ όσους χρησιμοποιούν καθημερινά τη γραμμή εντολών.

Αρχεία

Τα αρχεία παραμετροποίησης που μπορεί να χρειαστεί κάποιος για το zsh, τοποθετούνται είτε στο /etc είτε στον κατάλογο του χρήστη με πρόθεμα την τελεία. Τέτοια αρχεία είναι τα:

- zshenv
- zprofile
- zshrc
- zlogin
- zlogout

Στο αρχείο zcompdump τοποθετούνται όλα τα tab completions. Αν επιθυμεί κάποιος, μπορεί να συμπιέσει τα αρχεία αυτά μέσω της εσωτερικής εντολής zcompile, ώστε αυτά να αποθηκευτούν σε μία binary μορφή που έχει την κατάληξη .zwc, ώστε να φορτώνονται πιο γρήγορα. Το ιστορικό των εντολών κάθε χρήστη αποθηκεύεται στο αρχείο .zsh_history.

Globbing

Από τα πιο δυνατά σημεία του zsh είναι το globbing, δηλαδή, η διαδικασία με την οποία με βάση ένα όνομα ή αρχείο δημιουργείται μία λίστα αρχείων για χρήση/επεξεργασία από τη γραμμή εντολών. Globbing χρησιμοποιεί κάποιος, όταν, για παράδειγμα, θέλει να μετακινήσει ή να αντιγράψει ένα σύνολο αρχείων ή γενικώς να εκτελέσει εντολές σε ένα υποσύνολο αρχείων. Η χρήση τους μοιάζει αρκετά με τη χρήση regular expressions. Είναι πιο εύκολο να κατανοήσει κάποιος τη χρήση τους με παραδείγματα. Ετσι, για να πάρει κάποιος μία λίστα με όλα τα αρχεία που έχουν κατάληξη .c ή .o:

```
$ ls *.[co]
bar.c   bar.o   foo.c   inside.c linux.c  linux.o
```

Η παραπάνω εντολή τρέχει το ίδιο καλά και σε bash, ενεργοποιώντας, όμως, την επέκταση EXTENDEDGLOB του zsh, μπορούμε να φτιάξουμε πολύ πιο σύνθετα φίλτρα. Η ενεργοποίηση τέτοιων επεκτάσεων στο zsh γίνεται με την εντολή setopt.

```
$ setopt  EXTENDEDGLOB
$ ls ^*.c
bar.o    linux.h linux.o output
```

Με την παραπάνω εντολή ζητάει κάποιος όλα τα αρχεία που δεν έχουν κατάληξη .c.

Είναι επίσης δυνατόν να πάρει κάποιος μία λίστα με αρχεία που ταιριάζουν σε ένα εύρος από αριθμούς.

```
$ ls foo<20101101-20101103>.log
foo20101101.log foo20101102.log foo20101103.log
```

Από τις πιο χρήσιμες λειτουργίες του extended globbing είναι η αναζήτηση αρχείων σε υποκαταλόγους, χρησιμοποιώντας τα **, καθώς και του grouping μέσω των παρενθέσεων, αντικαθιστώντας έτσι τη χρήση της find σε ορισμένες περιπτώσεις.

```
$ ls **/(file|linux)*.c
dir1/file_in_dir1.c dir2/linux2.c       file1.c             file3.c
dir2/file_in_dir2.c dir3/file_in_dir3.c file2.c             linux.c
```

Η παραπάνω εντολή βρίσκει όλα τα αρχεία σε όλους τους υποκαταλόγους που ξεκινούν είτε με file είτε με linux και καταλήγουν σε .c.

Στο τέλος μίας εντολής αναζήτησης αρχείων μπορούν να προστεθούν και κάποια φίλτρα για τα δικαιώματα των αρχείων. (*) ή (x) για τα εκτελέσιμα, *(w) και *(r) για αρχεία με δυνατότητα εγγραφής ή ανάγνωσης αντίστοιχα.

```
$ ls *.c*(r)
bar.c    file1.c  file2.c  file3.c  foo.c    inside.c linux.c
$ chmod -r linux.c
$ ls *.c(r)
bar.c    file1.c  file2.c  file3.c  foo.c    inside.c
```

Φυσικά, όλα τα παραπάνω, καθώς και πολλά άλλα φίλτρα που υπάρχουν, μπορούν να συνδυαστούν, με σκοπό να πάρει κάποιος ακριβώς το αποτέλεσμα που ζητά.

Αντικατάσταση μέσα στη γραμμή εντολών

Στα περισσότερα shels μπορεί να γίνει αντικατάσταση της εξόδου μίας εντολής ως εισόδου σε μία άλλη μέσω των backticks (‘). Η κύρια και προτεινόμενη μέθοδος για να γίνει αυτό στο zsh, όμως, είναι το $(). Το πλεονέκτημα του $() είναι ότι δεν απαιτεί escaping για την περίπτωση που θα περιέχεται μία άλλη εντολή μέσα στην πρώτη. Για παράδειγμα, για να κάνει κάποιος strip όλα τα αρχεία που δεν έχουν γίνει μέχρι τώρα strip και περιέχονται σε ένα αρχείο με το όνομα file_list, με τη μέθοδο των backticks πρέπει να τρέξει:

```
$ strip `file \`cat file_list \`| grep 'not stripped' | cut -d:
 -f1`
```

με τη μέθοδο του $() αυτό γράφεται πολύ πιο απλά ως:

```
$ strip $(file $(cat file_list) | grep 'not stripped' | cut -d -
f1)
```

Αλλο ένα παράδειγμα χρήσης του $() που ανοίγει ένα vim με πολλά splitted παράθυρα, καθένα από τα οποία θα περιέχει ένα αρχείο .c από τους υποκαταλόγους.

```
$ vim -o $(ls **/*.c)
```

Ενας γρήγορος τρόπος να βρίσκει και να εκτυπώνει κάποιος την πλήρη διαδρομή ενός αρχείου που βρίσκεται στο $PATH του, είναι να χρησιμοποιεί το = ως πρόθεμα.

```
$ ls =sshd
/usr/sbin/sshd
```

Κάποιες φορές, στη γραμμή εντολών είναι χρήσιμο να αποθηκεύεται η έξοδος κάποιας εντολής μέσα στη γραμμή εντολών σε ένα προσωρινό αρχείο, το οποίο να μπορεί να χρησιμοποιηθεί ως είσοδος κάποιας άλλης. Αυτό γίνεται με τη χρήση του =().

```
$ diff =(ls dir1) =(ls dir2)
```

Εκτελώντας εντολές στο zsh.

```
1a2
> file_in_dir2.c
```

Αποθηκεύτηκε έτσι η έξοδος της ‘ls dir1’ σε ένα προσωρινό αρχείο και συγκρίθηκε μέσω της εντολής diff με την έξοδο της ‘ls dir2’. Μετά το τέλος της εκτέλεσης των εντολών, αυτά τα προσωρινά αρχεία σβήνονται αυτόματα. Με αυτόν τον τρόπο, είναι πολύ εύκολο να περάσει κάποιος την έξοδο μίας εντολής στον αγαπημένο του επεξεργαστή για περαιτέρω αλλαγές.

```
$ vi =(ps auxww | grep kargig)
```

Χρησιμοποιώντας έναν συνδυασμό από ^ μπορεί κάποιος να διορθώσει ένα τυπογραφικό λάθος από την προηγούμενη εντολή του.

```
$tail -f /tmp/lag_file
tail: cannot open `/tmp/lag_file' for reading: No such file
or directory
$ ^la^lo
$ tail -f /tmp/log_file
```

Δυστυχώς το ^xxx^yyy διορθώνει μόνο την πρώτη αναφορά του xxx όπου τη βρει και οι υπόλοιπες παραμένουν αναλλοίωτες. Για να επιτύχει κάποιος μαζική αλλαγή παραμέτρων, μπορεί να χρησιμοποιήσει την παρακάτω μέθοδο. Οπως και σε όλα τα shells, το !! επαναφέρει την προηγούμενη εντολή από το history, οπότε σε συνδυασμό με λίγο από regular expressions (προσοχή στο gs):

```
$ echo "foo 123 456 123"
foo 123 456 123"
$ !!:gs/123/999/
$ echo "foo 999 456 999"
foo 999 456 999"
```

Πολλές από τις δυνατότητες του zsh έχουν ενσωματωθεί στο Bourne shell 4, αλλά το zsh έχει ακόμη πολλούς άσσους στο μανίκι.

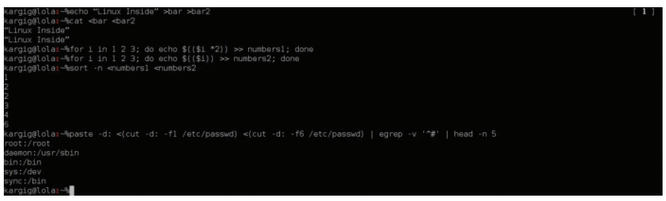
Ο editor της γραμμής εντολών του zsh λέγεται ZLE και μπορεί να χρησιμοποιηθεί είτε με keybindings είτε σε vi mode είτε σε emacs. Για να χρησιμοποιήσει κάποιος το vi mode, πρέπει να εκτελέσει:

```
'bindkey -v'
```

ενώ αντίστοιχα για emacs:

```
'bindkey -e'
```

Πατώντας PageUp ή Ctrl-P σε vi και emacs mode αντίστοιχα, μπορεί κάποιος να προσπελάσει τις προηγούμενες εντολές και να μετακινηθεί με τα βέλη δεξιά/αριστερά για να κάνει αλλαγές στην εντολή και να την τρέξει ξανά. Σε περίπτωση που έχει γράψει κάποιος μία μεγάλη εντολή, αλλά χρειάζεται κάτι ακόμη για να την ολοκληρώσει που δεν το θυμάται, αντί να σταματήσει την εντολή, αρκεί να πατήσει ESC-q (σε emacs mode) να γράψει μία νέα εντολή και μόλις αυτή εκτελεστεί, η γραμμή εντολών θα γυρίσει αυτόματα στην προηγούμενη



Εκτελώντας μερικές ακόμη εντολές στο zsh.

εντολή που έγραφε ο χρήστης. Χρησιμοποιώντας επίσης την επέκταση CORRECT, μπορεί κάποιος να έχει προτάσεις διόρθωσης για εντολές που έχει γράψει λάθος.

```
$ setopt CORRECT
$ chmdo u+x linux.c
zsh: correct 'chmdo' to 'chmod' [nyae]? y
```

Συμπλήρωση ονομάτων και εντολών με το TAB

Στα περισσότερα shells, πατώντας το πλήκτρο TAB, γίνεται αυτόματη συμπλήρωση ονομάτων αρχείων και εντολών έως το σημείο όπου το όνομα είναι μοναδικό. Στο zsh, αν πατήσουμε το πλήκτρο TAB για δεύτερη φορά, τότε εμφανίζεται μία λίστα με τα πιθανά ονόματα αρχείων που ταιριάζουν. Αν, όμως, ενεργοποιηθούν οι επεκτάσεις AUTO_LIST και AUTO_MENU, τότε κάθε φορά που πατά κάποιος το πλήκτρο TAB, γίνεται μία αυτόματη περιστροφή των πιθανών ονομάτων αρχείων που ταιριάζουν. Μπορεί επίσης να γίνει και επέκταση μεταβλητών περιβάλλοντος με το TAB.

Στο zsh, όμως, είναι εφικτός ο πλήρης προγραμματισμός της αυτόματης συμπλήρωσης μέσω της εντολής compctl. Σε περίπτωση που κάποιος θέλει στο javac να έχει ως input μόνο java προγράμματα από τον τρέχοντα κατάλογο και τους υποκαταλόγους του, μπορεί να τρέξει το παρακάτω:

```
$ compctl -g '*.java' + -g '*(-/)' javac
$ javac Javapr<TAB>
javac Javaprogs/test<TAB>
```

Global Aliases

Στο zsh, εκτός από τα τυπικά aliases, υπάρχουν και τα global. Με αυτά μπορεί κάποιος να εκτελέσει μία σειρά από αντικαταστάσεις κομματιών της γραμμής εντολών και όχι μόνο των εκτελέσιμων. Τυπικά παραδείγματα είναι να αντικαταστήσει κάποιος μία σειρά από εντολές που χρησιμοποιεί συχνά με κάποια, ένα συνήθως, γράμματα. Για παράδειγμα η εντολή:

```
$ sort blacklist.cf| uniq -c | sort -n | grep foo
```

μπορεί μέσω των

```
$ alias -g G='|grep'
$ alias -g UC='|uniq -c'
$ alias -g NS='|sort -n'
```

να μετατραπεί στην:

```
$ sort blacklist.cf UC NS G foo
```

Ακόμη ένα χρήσιμο παράδειγμα:

```
$ alias -g T10='|tail -n10'
$ cat foo T10
```

Είναι φανερό το πόσο πρακτικό είναι κάτι τέτοιο για όσους χρησιμοποιούν συχνά το τερματικό. Τα global aliases μπορούν να γλιτώσουν πολύ χρόνο σε πληκτρολόγηση, αλλά η επιλογή των ονομάτων τους θέλει προσοχή, για να μην προκύψουν προβλήματα με ονόματα αρχείων, διεργασιών κ.ά.

```
javac Javaprogs/test.java
```

Χρησιμοποιώντας τις δυνατότητες του compctl σε συνδυασμό με functions, μπορεί κάποιος να έχει αυτόματη συμπλήρωση ακόμη και των επιλογών μίας εντολής. Για παράδειγμα:

```
$ mount -<TAB>
option
-a  — mount all filesystems in fstab
-f  — fake mount
-o  — specify file system options
-t  — specify file system type
```

Αν θέλουμε να δημιουργήσουμε μία λίστα με πιθανές επιλογές για μία εντολή, δεν έχουμε παρά να χρησιμοποιήσουμε την εντολή compctl -k.

```
$ compctl -k '(pull push status diff)' git
$ git <TAB>
diff    pull    push    status
```

Για να παραμετροποιήσουμε το auto-completion του zsh, χρησιμοποιούμε την εντολή 'zstyle' στο zshrc του. Μπορούμε να ενεργοποιήσουμε την cache, ώστε να έχουμε πιο γρήγορα αποτελέσματα για τη συμπλήρωση εντολών με πολύ μεγάλη έξοδο, π.χ., τα πακέτα προς εγκατάσταση μέσω του apt, ή ακόμη να ορίσουμε το completion να είναι case insensitive για τα ονόματα που αρχίζουν με πεζά, αλλά case sensitive για τα κεφαλαία:

```
zstyle ':completion:*' use-cache on
zstyle ':completion:*' cache-path ~/.zsh/cache
zstyle ':completion:*' matcher-list 'm:{a-z}={A-Z}'
```

Αν δεν θέλει κάποιος τα αρχεία που θα εμφανίζονται στο completion να είναι ταξινομημένα με σειρά ονόματος, αλλά προτιμά να είναι με τη σειρά που τα έχει προσπελάσει τελευταία, αρκεί να αλλάξει τη μέθοδο file-sort. Για completion σε

> **Οι δυνατότητες αυτόματης συμπλήρωσης του zsh είναι πραγματικά πολύ εκτεταμένες και το πακέτο εγκατάστασης του zsh έρχεται με πάρα πολλές προεπιλογές, που είναι στο χέρι του χρήστη να τις ενεργοποίησει.**

εντολές που απαιτούν hostnames (π.χ., ssh), μπορεί κάποιος να δημιουργήσει αρκετά εύκολα μία λίστα με πιθανά hostnames από διάφορα αρχεία στον υπολογιστή του και το zsh να τα χρησιμοποιήσει μέσω της μεθόδου 'hosts':

```
zstyle ':completion:*' file-sort access
hosts=( $(cat /etc/hosts | grep -v "^#" | awk '{print $1}'|
 cut -d"," -f1), \
$(cat $HOME/.ssh/known_hosts | grep -v "^|1|" | awk
'{print $1}'| cut -d"," -f1))
zstyle ':completion:*' hosts $hosts
```

Τα παραδείγματα μπορούν, φυσικά, να γίνουν ακόμη πιο σύνθετα. Αν θέλει κάποιος να αγνοεί πάντα τα αρχεία με συγκεκριμένες καταλήξεις από τις εντολές του, εκτός αν αυτή η εντολή είναι η rm, δεν έχει παρά να χρησιμοποιήσει το παρακάτω zstyle:

```
zstyle ':completion:*:*:(^rm):*:*files' ignored-patterns \
'*?.(c~|old|zwc|bak)' '*~'
```

Ενα πολύ χρήσιμο τρικ για το auto-completion είναι η εμφάνιση ενός menu με τις πιθανές επιλογές μέσα στο οποίο μπορεί κάποιος να κινηθεί με τα βέλη. Για να εμφανιστεί ένα τέτοιο menu με τα πιθανά process Ids κατά τη χρήση της εντολής kill, αρκεί η προσθήκη των παρακάτω:

```
zstyle ':completion:*:*:kill:*' menu yes select
zstyle ':completion:*:kill:*'   force-list always
```

Για μία λίστα με όλα τα zstyles που είναι ενεργοποιημένα μπορεί κάποιος να εκτελέσει την εντολή 'zstyle -L', ενώ περισσότερα για τη διαμόρφωση των zstyles μπορεί κάποιος να διαβάσει στο manpage zshmodules. Οι δυνατότητες αυτόματης συμπλήρωσης του zsh είναι πραγματικά πολύ εκτεταμένες και το πακέτο εγκατάστασης του zsh έρχεται με πάρα πολλές προεπιλογές, που είναι στο χέρι του χρήστη να τις ενεργοποίησει. Η τεκμηρίωση του zsh πάνω στο συγκεκριμένο θέμα είναι εκτεταμένη και για να καλύψει κάποιος το θέμα του completion στο zsh, θα χρειαζόταν σίγουρα ένα ξεχωριστό άρθρο.

Πολλαπλές ροές εισόδου/εξόδου

Στο zsh είναι δυνατόν να χρησιμοποιήσει κάποιος πολλαπλές ροές εισόδου και εξόδου για ένα πρόγραμμα.

```
$ echo "Linux Inside" >bar >bar2
$ cat <bar <bar2
Linux Inside
Linux Inside
```

Φυσικά, είναι εφικτό να επεξεργαστεί κάποιος και την έξοδο από πολλαπλές ροές. Παραδείγματα:

```
$ for i in 1 2 3; do echo $(($i *2)) >> numbers1; done
$ for i in 1 2 3; do echo $(($i)) >> numbers2; done
$ sort -n <numbers1 <numbers2
1
2
2
3
4
6

$ paste -d: <(cut -d: -f1 /etc/passwd) <(cut -d: -f6
/etc/passwd) | egrep -v '^#' | head -n 5
root:/root
daemon:/usr/sbin
bin:/bin
sys:/dev
sync:/bin
```

Στο bash, για να εξαγάγει κάποιος το αποτέλεσμα μία εντολής και σε ένα αρχείο και στο standard output, χρησιμοποιεί την εντολή tee. Στο zsh αυτό δεν χρειάζεται και μπορεί κάποιος να χρησιμοποιήσει ένα απλό pipe.

```
bash $ ls | tee output.log
zsh $ ls >output.log | cat
```

Στο Internet θα βρει κάποιος functions για το RPROMPT που μπορούν να μπουν στο .zshrc, τα οποία δείχνουν ημερομηνία/ώρα, την κατάσταση σε ένα git repository, τη φόρτιση της μπαταρίας κ.ά.

Χρήσιμες επεκτάσεις

Οπως έχει ήδη ειπωθεί, οι επεκτάσεις του zsh ενεργοποιούνται μέσω της εντολής setopt. Μία πάρα πολύ χρήσιμη επέκταση του zsh είναι η 'cdable_vars', με την οποία είναι εφικτό να αναθέσει κάποιος σε μία παράμετρο ένα πολύ μεγάλο path και έπειτα να εκτελέσει την εντολή cd με όρισμα αυτή την παράμετρο και να μεταφερθεί αυτόματα στο παραπάνω path. Ακόμη μία χρήσιμη επέκταση σχετική με καταλόγους είναι η 'auto_cd'. Ενεργοποιώντας την, αρκεί κάποιος να δώσει στο terminal το path όπου επιθυμεί να μεταβεί, χωρίς την εντολή 'cd', και το zsh θα τον μεταφέρει αυτόματα εκεί. Με την επέκταση 'globdots' μπορεί να γίνει επέκταση παραμέτρων και επάνω στα αρχεία που αρχίζουν με '.'. Η ενεργοποίηση της επέκτασης 'no_clobber' προστατεύει τα αρχεία με μέγεθος μεγαλύτερο του μηδενός από να χαθούν (να «αδειάσουν») μέσω μίας λάθος ανακατεύθυνσης με το '>'. Για να γίνει όντως ανακατεύθυνση επάνω τους, χρειάζεται να χρησιμοποιηθεί το '>!'. Σε περίπτωση που δεν γίνονται συχνές αλλαγές στα path των εκτελέσιμων εντολών, η επέκταση 'hash_cmds' είναι αρκετά χρήσιμη, μια και αποθηκεύει το path της εντολής και η επόμενη κλήση της εντολής θα γίνει χωρίς να χρειαστεί αναζήτηση μέσα στο $PATH. Ενα από τα πιο σπουδαία και χρήσιμα χαρακτηριστικά του zsh είναι η δυνατότητά του να μοιράζεται το history μεταξύ πολλαπλών sessions. Με την επέκταση 'append_history' όλα τα sessions θα προσθέτουν το history τους στο history file, αντί να γράφουν από πάνω του, ενώ ενεργοποιώντας την επέκταση 'share_history', κάθε εντολή από κάθε session μπαίνει κατευθείαν στο history file προτού τελειώσει το session. Ετσι, μια εντολή που έχει τρέξει από ένα session, μπορεί να αναζητηθεί και να κληθεί από ένα άλλο, χωρίς να χρειαστεί να κλείσει πρώτα το πρώτο session.

```
kargig@lola:~%kill 2828
process ID
 2768 ?        00:02:00 fluxbox
 2814 ?        00:00:01 ssh-agent
 2817 ?        00:00:00 dbus-launch
 2818 ?        00:00:03 dbus-daemon
 2822 ?        00:00:04 gnome-power-man
 2826 ?        00:00:48 xbindkeys
 2828 ?        00:00:06 halevt
 2831 ?        00:00:00 gconfd-2
 2903 ?        00:00:00 gvfsd
 3960 ?        00:00:00 gvfsd-trash
 3962 ?        00:00:00 gvfs-gdu-volume
 3967 ?        00:00:00 gvfs-gphoto2-vo
 3969 ?        00:00:03 gvfs-afc-volume
 6274 ?        00:00:00 gvfsd-metadata
 9288 ?        00:00:00 swiftfox
```

Με χρήση της zstyle μπορούμε στο auto-completion να εμφανίζουμε μενού με τις πιθανές επιλογές στις οποίες μπορούμε να κινούμαστε με τα βελάκια στο πληκτρολόγιο

> Στο δίκτυο υπάρχουν πολλά zshrc για να πάρετε ιδέες. Το δικό μου μπορείτε να το βρείτε στο http://goo.gl/vOnMm.

Οι δυνατότητες του zsh είναι πραγματικά απεριόριστες και όσο ασχολείται κάποιος μαζί του τόσο αυτό κάνει πιο εύκολη τη δουλειά του και τη ζωή του. Η μετάβαση από το bash είναι αρκετά εύκολη και σίγουρα το zsh δεν θα απογοητεύσει όποιον του αφιερώσει τον απαραίτητο χρόνο. chsh -s zsh!

Σύνδεσμοι

Zsh: http://goo.gl/1e7t7
Gentoo Zsh Howto: http://goo.gl/OF6nV
Zsh Tips: http://goo.gl/19iP8
Zsh-lovers: http://goo.gl/VlLMp
zshrc: http://goo.gl/vOnMm